Dremel: Interactive Analysis of Web-Scale Datasets

CS 744 BIG DATA
PHIL MARTINKUS
Motivation

Large Scale Data must be accessible to analysts and engineers

Interactive queries are important for data exploration, monitoring, online customer support, rapid prototyping, debugging of data, and other tasks

Many databases require a costly loading phase

Web data is often non-relational
The Solution: Dremel

Dremel is a system that supports interactive analysis of very large datasets over shared clusters of commodity machines.

Has been in production at Google since 2006

Can operate on data in place using a distributed storage system

Uses a novel columnar storage format for nested data

Provides a high-level SQL-like language for interactive queries
Data Model

Strongly-typed nested records

Records consist of one or more fields

Fields can be required, optional or repeated
Data Model Example

Each Record represents a document

Required DocId field

Links is an optional group with two nested repeated fields.

Name is a repeated group with a nested Language group.
Nested Columnar Storage

All values of a field are stored consecutively in blocks

Goals for the storage system:
- Lossless representation of record structure in columnar format
- Fast encodings
- Efficient record assembly

Repeated records are handled with repetition and definition levels
Repetition Levels

Used to disambiguate occurrences of the same field within the same record.
Tell us at what repeated field in the field's path the value has repeated.
Repetition Level Example
Definition Levels

Whenever an optional or repeated field is not present in a record, the system stores a NULL.

Tell us how many fields in the field's path that could be undefined (because they are optional or repeated) are actually present in the record.

Mostly useful for distinguishing NULL values.
## Definition Level Example

**message Document**

```java
message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward;
  }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country;
      optional string Url;
    }
  }
}
```

<table>
<thead>
<tr>
<th>DocId</th>
<th>Name.Url</th>
<th>Links.Forward</th>
<th>Links.Backward</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td><a href="http://A">http://A</a></td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td><a href="http://B">http://B</a></td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td><a href="http://C">http://C</a></td>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name.Language.Code</th>
<th>Name.Language.Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>en-us</td>
<td>us</td>
</tr>
<tr>
<td>en</td>
<td>NULL</td>
</tr>
<tr>
<td>NULL</td>
<td>gb</td>
</tr>
<tr>
<td>en-gb</td>
<td>0</td>
</tr>
<tr>
<td>en</td>
<td>0</td>
</tr>
<tr>
<td>NULL</td>
<td>0</td>
</tr>
<tr>
<td>NULL</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DocId</th>
<th>Name</th>
<th>Links.Backward</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td><a href="http://A">http://A</a></td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td><a href="http://B">http://B</a></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td><a href="http://C">http://C</a></td>
<td>80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DocId</th>
<th>Name</th>
<th>Links.Backward</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td><a href="http://A">http://A</a></td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td><a href="http://B">http://B</a></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td><a href="http://C">http://C</a></td>
<td>80</td>
</tr>
</tbody>
</table>
Splitting Records into Columns

Recursive algorithm computes levels for each field
A tree of field writers match the structure of the field schema
Many datasets at Google are sparse
Record Assembly

Goal is to reconstruct records given a subset of fields

Finite state machine (FSM) reads values and appends to output records

An FSM state corresponds to a field reader

The FSM is traversed from the start state to the end state for each record
Query Language

Based on SQL

Designed for columnar nested storage
Query Execution

Uses a Tree architecture
Root receives incoming queries
Intermediate servers rewrite the query
Leaf servers access data
Each server has an internal tree corresponding to a physical query execution plan.
Query Execution Example

A set of iterators scan the input column in lockstep and emit results with annotated repetition and definition levels without actually assembling the records.
Query Dispatcher

Dremel is a multi-user system that executes queries simultaneously

The query dispatcher schedules queries

Dealing with stragglers

- Disproportionally slow processes are rescheduled on another server
- A parameter specifies the minimum percentage of tablets that must be scanned before returning a result
Experiments
Questions?